



LABORATÓRIO DE BANCO DE DADOS

IBD100

SQL – Structured Query Language

- SQL – Comandos, comandos e comandos...
 - Já são conhecidas a origem e a evolução da SQL...
 - Já se entende como e porque os SGBDs adotam a SQL como linguagem...
 - Para analistas e programadores é muito interessante aprender somente uma linguagem e poder trabalhar com diversos gerenciadores de banco de dados.
 - Agora serão explorados os comandos da SQL usando o ambiente do pgAdmin 4 para acessar e 'rodar' os comandos no SGBD PostgreSQL.
 - Vale lembrar: SQL é tecnologia inicialmente desenvolvida para implementar a AR. MAS... com o aumento das necessidades, acabou indo além do que a AR propunha. Por isso, houve este 'descolamento' da SQL em relação à AR.
 - Quando a SQL se iniciou, diziam que a linguagem era aderente à AR. Entretanto, com a evolução da SQL, a 'aderência' não se manteve.

- SQL – Comandos, comandos e comandos...
 - A 'NÃO - aderência' da SQL com a AR começou quando a tecnologia começou a oferecer operações de processamento de dados que a AR não tinha.
 - Exemplo: operador JOIN possibilitando fazer junções abertas. 'Necessidade' atendida e que mais tarde 'forçou' uma revisão da AR.
 - Outra situação que a tecnologia propiciou e a AR não contemplava combinação de comandos de SQL.
 - Em SQL, como se vê, é possível (e muitas vezes interessante) realizar uma projeção, uma seleção e uma junção em somente um comando. Isso em AR deveria ser escrito em três operações.
 - Então a SQL é mais 'completa' que a AR? Sendo simples: Sim.

- SQL – Comandos, comandos e comandos...
 - A 'NÃO - aderência' da SQL com a AR começou quando a tecnologia começou a oferecer operações de processamento de dados que a AR não tinha.
 - Exemplo: operador JOIN possibilitando fazer junções abertas. 'Necessidade' atendida e que mais tarde 'forçou' uma revisão da AR.
 - Outra situação que a tecnologia propiciou e a AR não contemplava combinação de comandos de SQL.
 - Em SQL, como se vê, é possível (e muitas vezes interessante) realizar uma projeção, uma seleção e uma junção em somente um comando. Isso em AR deveria ser escrito em três operações.
 - Então a SQL é mais 'completa' que a AR? Sendo simples: Sim.
 - Os desenvolvedores dos diferentes SGBDs perceberam que a evolução do SQL não dependia do desenvolvimento da AR. E ficaram 'meio que na terra de ninguém', e começaram a escrever funções em seus produtos

- SQL – Comandos, comandos e comandos...
 - Então o desenvolvimento tecnológico da SQL aumentou a NÃO-Aderência com a AR e acelerou a diversificação de 'linguagens quase-SQL' implementadas nos SGBDs? Sim.
 - Daí o ANSI acelerar as padronizações, lembrando: 1987, 1989 [atualização da 87], 1992 [a mais usada nos SGBDs], 1999, 2003, 2006, 2008, 2011, 2016 e 2019, mas de 92 em diante a SQL descolou de tal forma que a padronização 'correu atrás' do padrão de mercado ou da tendência de mercado.
 - Hoje os SGBD tem 3 'categorias' de comandos em relação à SQL:
 - SQL-Padrão [em geral ANSI-92].
 - Nativa [comandos do SGBD que nada tem a ver com SQL: $\exists$ comando find ou seek para fazer seleção, p.e.].
 - SQL-Nativo [parece a SQL, mas executando funções específicas de um SGBD].

- Antes de começar com a SQL, serão restaurados conceitos e fundamentos de BD e de SGBD.

Tópico	Definição
Banco de Dados	Conjunto de arquivos que armazenam os dados de uma organização
Base de Dados	Conjunto de arquivos de dados e também todos os outros componentes que dão suporte ao funcionamento de um sistema de informação
Tabela	Representação de um dos conjuntos de dados necessários para definir o banco de dados de uma organização. A tabela deve ser correspondente a um conjunto no sentido estrito da matemática. Ou seja, deve conter dados de um só conjunto importante para a realização dos procedimentos e controle das regras de uma organização
Tupla	Também denominada LINHA da tabela, é uma coleção de valores que representam uma ocorrência da realidade que está retratada na tabela
Campo	Uma das colunas de uma tabela. De outra forma, é a propriedade associada a um grupo de valores contidos em uma tabela, assim sendo, podemos dizer que, em uma tabela de PRODUTOS, um campo seria a descrição do produto ou outro campo seria a quantidade em estoque
Domínio de valor	Regra que define um intervalo de valores válidos para um campo da tabela.

- Conceitos iniciais de BD e SGBD

Tópico	Definição
Chave Primária	Um ou mais de um campo de uma tabela, tal que, fornecidos seus valores, localiza-se na tabela, no máximo, uma tupla.
Chave Estrangeira	Um ou mais de um campo de uma tabela, tal que, seus valores indicam tuplas únicas em uma outra tabela (aponta para valores de chave primária).
Chave Candidata	Um ou mais de um campo de uma tabela que pode candidata a ser a Chave Primária da Tabela.

- Notas:
- A Chave Primária é uma consequência do Princípio da Unicidade da Tupla do MR.
- As Chaves Estrangeiras deveriam seguir a 2ª Regra de Integridade do MR que diz: Os valores das Chaves Estrangeiras devem obedecer à regra:
 - Serem NULOS ou
 - Indicarem um valor que EXISTA na sequência de valores da Chave Primária correspondente.
- Os SGBDs 'Aderentes ao MR' devem seguir os princípios e as Regras de Integridade.

- Conceitos iniciais de BD e SGBD

- Princípios do MR

- As tabelas não devem apresentar campos repetidos em sua estrutura (esquema).
 - As linhas de $\forall$ tabela não podem apresentar repetição. Isto é a UNICIDADE da TUPLA.
 - A ordem dos campos no esquema é irrelevante
 - A ordem das linhas na tabela é irrelevante

- Regras de Integridade:

- 1ª – Toda tabela deve ter sua chave primária.
 - 2ª – Toda Chave estrangeira pode ter somente um dos dois valores:
 - Ser NULO – Não existe valor
 - Ter uma dos valores que estão na correspondente Chave Primária.

- Os SGBD são o resultado do amadurecimento das necessidades empresariais para suporte no desenvolvimento de Sistemas de Informação
- OS SGBD devem dar suporte ao projeto do SI, portanto devem permitir:
 - Criar os arquivos que armazenam os dados e administrar estes arquivos
 - Implementar os programas aplicativos
- Os SGBDs têm comandos que permitem 'operar' diferentes funcionalidades. Estes comandos são agrupados em:
 - DDL
 - Comandos para definir a existência de componentes de uma Base de Dados
 - DCL
 - Comandos que permitem definir usuários e grupos de usuários e configurar seus privilégios de acesso aos componentes de uma base de dados.
 - DML
 - Comandos que permitem operar (manipular ou atualizar) os dados contidos nos componentes

- Quais comandos SQL serão vistos?
- Comandos que implementam as operação de AR...

- SELEÇÃO
- PROJEÇÃO
- PRODUTO CARTESIANO
- JUNÇÃO Natural
- JUNÇÃO Aberta Pela Esquerda
- JUNÇÃO Aberta Pela Direita
- JUNÇÃO Completa
- UNIÃO
- INTERSECÇÃO
- SUBTRAÇÃO

$R \leftarrow A \text{ [[Condição]]}$

$R \leftarrow A \text{ [[C1, C2,..., Cn]]}$

$R \leftarrow A \times B$

$R \leftarrow A \text{ [[Condição]] } B$

$R \leftarrow A \text{ [[Condição]] } B$

$R \leftarrow A \text{ [[Condição]] } B$

$R \leftarrow A \text{ [[Condição]] } B$

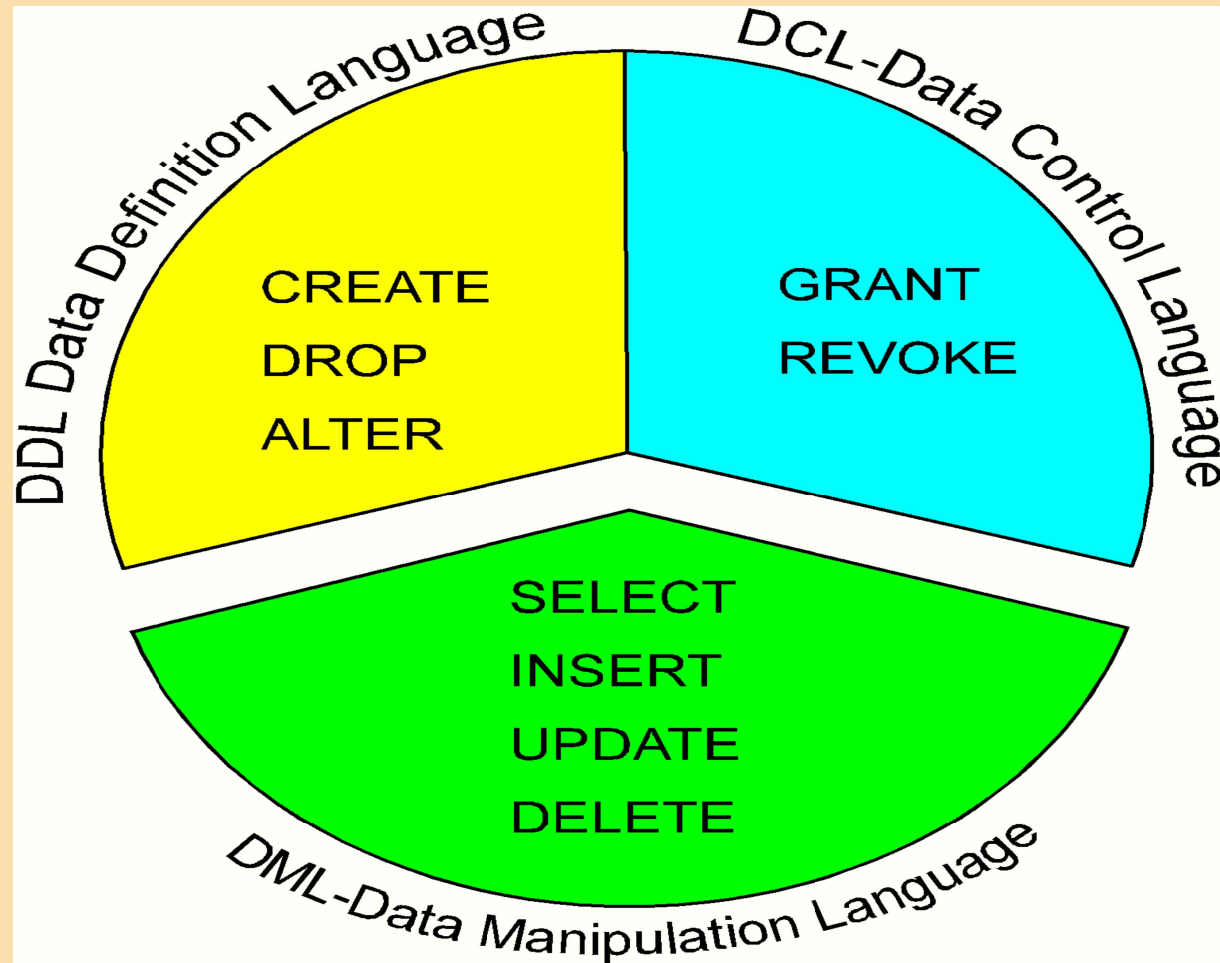
$R \leftarrow A \cup B$

$R \leftarrow A \cap B$

$R \leftarrow A - B$

- E mais os comandos de atualização das tuplas das tabelas...
 - Inclusão
 - Alteração
 - Exclusão

- Os comandos da SQL e os Grupos de Comandos dos SGBDs são os seguintes:



- Os Comandos de DML de Consulta são implementados em SQL com o comando, o SELECT, com a sintaxe:
- ```
SELECT [ALL | DISTINCT [ON (expressao[,...])]]
 * | Campo1[AS nome_saída1][,...]
FROM TAB1[, TAB2, TAB3, ... , TABn]
[WHERE condição]
[GROUP BY chave1[,chave2,...]]
 [HAVING condição[,...]] [{ } select]
[ORDER BY campo1 [ASC|DESC|USING operador] [,
 campo2 [ASC|DESC|USING operador] [,
 ...campon] ;
```
- Tudo o que está entre [] é opcional
- O comando SELECT implementa: Seleção, Projeção; para o Produto Cartesiano, a Junção Natural, as Junções Abertas (Direita, Esquerda e Completa), a União, a Subtração e a Intersecção são usados *operadores*.
- As funções podem ser escritas na operação de Projeção.
- Os complementos de comandos de ordenação podem apresentar mais de um campo de ordenação (indicando subordenações)
- Os agrupamentos podem apresentar o uso de funções para submissão de seleção de dados (no agrupamento)

- Os Grupos de comandos podem ser definidos ...

| Tópico           | Definição                                                                                                                                                                                                                                                                                             |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SELECT           | Indica ao núcleo que deverá ser iniciado o processamento de leitura de dados das tabelas                                                                                                                                                                                                              |
| *  campo [as c1] | Indica quais campos devem ser exibidos no resultado do comando. Se usarmos * vemos todos os campos processados, senão devemos indicar quais queremos ver. O uso do "as" permite o renomear os campos dinamicamente.                                                                                   |
| FROM             | Especifica a(s) tabela(s) a processar. No caso de mais de uma tabela, os nomes devem ser separados por ','                                                                                                                                                                                            |
| WHERE            | Condição do processamento das linhas da tabela. Vale combinar condições com operadores lógicos [AND, OR, NOT, XOR]. Os operadores de comparação Padrão ANSI/99 são: =, <>, >, <, >=, <=, IN, BETWEEN; além dos específicos de cada SGBD.                                                              |
| GROUP BY         | Permite estabelecer agrupamentos de valores que podem apresentar a execução de funções escalares ou aritméticas                                                                                                                                                                                       |
| HAVING           | Executam funções sobre valores de campos, podem indicar resultados que podem ser usados como parâmetro de agrupamento.                                                                                                                                                                                |
| ORDER BY         | Classificam os dados das tuplas segundo campos da tabela permitindo estabelecer organizações para execução de funções. As chaves de ordenação podem apresentar sub-chaves de ordenação e, para cada uma delas, ter ordens diferentes indicadas pelo uso de ASC ou DESC seguindo cada chave declarada. |



- SELEÇÃO
- Da AR... <tab> [[ Condição ]]
  - Exemplo: funcionarios [[ cedepo='A01' ou cedepo='E11']]
- Em SQL

```
select * from funcionarios where cedepo='A01' or cedepo='E11';
```
- Notas
  - As linhas são recuperadas na ordem como estão gravadas no arquivo (da tabela funcionarios). Isso é notado SE este comando for executado sem a existência da Chave Primária (CP)
  - Vários SGBDs implementaram um operador equivalente ao 'or', na forma:

```
select * from funcionarios where cedepo in('A01','E11');
```
- É possível 'criar' uma tabela com os dados processados usando o comando...

```
create table R as (select * from funcionarios where cedepo in('A01','E11'));
```
- Ou ainda...

```
create temporary table R as (select * from funcionarios where cedepo in('A01','E11'));
```

Nesta segunda forma a tabela fica referenciada em memória. Tendo os acessos seguintes feitos com melhor performance.
- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>

- PROJEÇÃO

- Da AR... <tab> [[ Lista de Campos de tab ]]

- Em SQL

```
select idfuncionario, txprenomes,txsobrenome,cedeppto,dtnascimento from funcionarios;
```

- Notas

- A lista de linhas na resposta deste comando tem a mesma quantidade de linhas da tabela processada.

Considere uma projeção mais simples: `select txsobrenome from funcionarios;`

Mesmo neste comando a quantidade de linhas na resposta será igual que a da tabela processada.

- A Não-Aderência da SQL com a AR permite combinar a Seleção e a Projeção em só comando, assim:

```
select idfuncionario, txprenomes,txsobrenome,cedeppto,dtnascimento
from funcionarios
where cedeppto='A01' or cedeppto='E11';
```

- Note: não é só uma seleção, não é só uma projeção. Não tem 'tradução' para AR, mas faz seleção e projeção.

- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>

- PRODUTO CARTESIANO
- Da AR... <tab1> X <tab2>
- Em SQL [ANSI-92] se usa o operador para todos os 'emparelhamentos' de linhas de tabelas (ou de conjuntos). Na SQL [ANSI-87] não existe a operação de junções abertas.
- O operador é o JOIN, mas existem 5 formas de emparelhamentos (não lembra? Consulte os slides de AR).
- Então, o JOIN tem 5 complementos. Para o Produto Cartesiano, o complemento é CROSS, pode-se, então, escrever:  
Sintaxe SQL: `select * from <tab1>, <tab2>;` - ANSI-87  
`select * from <tab1> CROSS JOIN <tab2>;` - ANSI-92
- Exemplo:  
`select * from setoresdeatuacao, planosdesaude; -- ANSI-87`  
`select * from setoresdeatuacao CROSS JOIN planosdesaude; -- ANSI-92`
- -- combinando comandos  
`select cpsetoratuacao, txnomesetordeatuacao, cpplanodesaude, txnomeplano  
from setoresdeatuacao CROSS JOIN planosdesaude  
where txnomeplano in ('Unimed', 'SulAmericana');`
- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>

- JUNCÃO NATURAL
- Da AR... <tab1> [[ Condição ]] <tab2>
- Em SQL existem dois modos fundamentais para escrever este comando.

Sintaxe:

**ANSI-87** select \* from <tab1>, <tab2> where <condição>;

**ANSI-92** select \* from <tab1> INNER JOIN <tab2> ON <condição>;

- Exemplos:

```
select * from funcionarios, departamentos where cedepto=cpdepto;
```

ou

```
select * from funcionarios INNER JOIN departamentos ON cedepto=cpdepto;
```

ou

```
select * from funcionarios INNER JOIN departamentos
 ON funcionarios.cedepto=departamentos.cpdepto;
```

ou

```
select * from funcionarios as f INNER JOIN departamentos as d
 ON f.cedepto=d.cpdepto;
```



- **JUNÇÃO NATURAL**

- **Notas:**

- Em muitos gerenciadores, SE os campos envolvidos na <condição> tiverem o MESMO nome, é possível escrever:

`select * from <tab1> inner join <tab2> using <campo>;`

MAS, veja que isso DEPENDE do modelo lógico, que é onde se define os nomes dos atributos

Na base de dados do PraticaSQL não existem tabelas com Chave Primária e Estrangeira com mesmo nome.

Entretanto, é possível alterar a estrutura de uma tabela e testar o comando alternativo.

Escreva:

```
alter table if exists servicosprestados rename ceveiculo to cpveiculo;
```

E a seguir:

```
select * from veiculos inner join servicosprestados using cpveiculo;
```

Depois de testar o comando temos que alterar a estrutura da tabela com o comando

```
alter table if exists servicosprestados rename cpveiculo to ceveiculo;
```

- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>



- JUNÇÃO ABERTA PELA DIREITA | ESQUERDA | COMPLETA
- Da AR... <tab1> [[ Condição ]<tab2> OU <tab1> ]<tab2> OU <tab1> ]<tab2>
- Em SQL...
- Na base de dados do PraticaSQL as tabelas medicos e instituicoesdeensino já tem dados para teste dos três comandos.
- Os comandos abaixo combinam a projeção com as junções abertas [DIR, ESQ e COMPL]

```
select cpmedico, txnomemedico, nucrm, cpinstituicao, txnomeinstituicao, dtfundacao
from instituicoesdeensino right join medicos on cpinstituicao=ceinstituicao;
```

```
select cpmedico, txnomemedico, nucrm, cpinstituicao, txnomeinstituicao, dtfundacao
from instituicoesdeensino left join medicos on cpinstituicao=ceinstituicao;
```

```
select cpmedico, txnomemedico, nucrm, cpinstituicao, txnomeinstituicao, dtfundacao
from instituicoesdeensino full outer join medicos on cpinstituicao=ceinstituicao;
```

- É possível gerar dados em outras tabelas (com CPs e CEs com valores certos) para testar os comandos das junções abertas. Mas gerar estes dados veremos os comandos de atualização de dados.
- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>

- COMANDOS DE ATUALIZAÇÃO - INSERT

- Faz a inclusão de uma ou mais linhas em UMA tabela.

- Sintaxe:

```
- INSERT INTO <Tab> [(Campo1, Campo2, ...)]
 VALUES ('valor do campo1', 'valor do campo 2', ...) [,
 ('valordocampo 1', 'valordocampo 2', ...),
 ('valordocampo 1', 'valordocampo 2', ...),
 ('valordocampo 1', 'valordocampo 2', ...)];
```

- Notas

- Geral: Os comandos de Atualização atuam somente sobre uma tabela. Isso pode ser uma limitação da SQL que foi resolvida parcialmente em poucos SGBDs.
- Se a lista de campos depois de <tab> for omitida, os valores (após *values*) devem seguir a ordem de campos que está definida na tabela.
- A verificação da consistência de valores é feita pelo módulo de verificação de integridade do SGBD. Esta verificação pode gerar uma situação de erro na atualização do dado e neste caso, a NÃO execução do comando.
- Mais de uma linha pode ser inserida em um mesmo comando. Note a lista de valores entre [ ].

- **COMANDOS DE ATUALIZAÇÃO - INSERT**

- **Exemplo**

- ```
insert into departamentos
    (cpdepto,txnomedepcto,cefuncionariogerente,cedeptosuperior,
     txlocalizacao,qtareatotal,dtcriacaodepto,dtcaddepartamento)
values
    ('G01','Desenvolvimento de sistemas embarcados','10','A01','Rua Direita,450',1500,'2025-03-01','2025-03-23');
```

- **Notas**

- Os dados a serem inseridos podem ser resultado de comandos SQL, supondo existir uma tabela com nome funcsbeneficiarios e com a mesma estrutura de funcionarios, então se escreve na forma...

```
insert into funcsbeneficiarios values
    (select * from funcionarios where round(((current_date-dtcontratacao)/365.25),2) > 30 );
```

- No comando anterior diz-se que o comando entre () está *aninhado* ou é um *Sub-Comando* ou ainda uma *Sub-Query*.
 - Existe ainda uma sintaxe do insert que vale para alguns SGBDs (mas não é ANSI)...
- `insert into <tab> SET <c1>=<v1>, <c2>=<v2>,...,<cN>=<vN>;` - c1,...,cN são campos de <tab> e v1,...,vN são os valores atribuídos aos campos. Entretanto, esta sintaxe NÃO é executada no PostgreSQL.

- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>

- **COMANDOS DE ATUALIZAÇÃO - DELETE**

- Faz a exclusão de uma ou mais linhas de uma tabela.

- **Sintaxe:**

`DELETE FROM <Tab> [WHERE <condição>];`

- **Exemplo**

`Delete from cursos where ceinstdeensino in ('1','3');`

- **Notas**

- Se a parte WHERE for omitida TODAS AS LINHAS de <tab> serão excluídas.
- A maior parte dos SGBD aumenta o espaço ocupado pelo arquivo que armazena os dados. MAS NÃO DIMINUI o tamanho do arquivo quando linhas são excluídas.

Por que?

Existem muitas respostas para esta pergunta... a mais tranquila: porque é difícil de implementar.

- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>

- **COMANDOS DE ATUALIZAÇÃO – UPDATE**

- Faz a alteração de valores em campos de uma tabela.

- Sintaxe

- Update <tab> SET <campo1>=<valor1>[,<campo2>=<valor2>,...,<campoN>=<valorN>]
[where <condição>];

- Exemplo

- Update funcionarios set aosituação='B' where idfuncionario in ('1','3');

- Notas

- Este comando atua somente em uma tabela. Entretanto, existem SGBDs que conseguem fazer junções a proporcionar alterações nos registros de uma das tabelas relacionadas. Escrevendo comandos semelhantes:

- Update nfvendas inner join nfitensvendas on nfvendas.cpnunfvenda=nfitensvendas.cenunfvenda
set nfvendas.vltotalnota=SUM(nfitensvendas.vlunititem*nfitensvendas.qtitem);
 - O PostgreSQL implementa algo parecido. Pesquise.

- Mais comentários, dicas de uso e sintaxes complementares acesse <https://www.postgresql.org/docs/>



LABORATÓRIO DE BANCO DE DADOS

IBD100